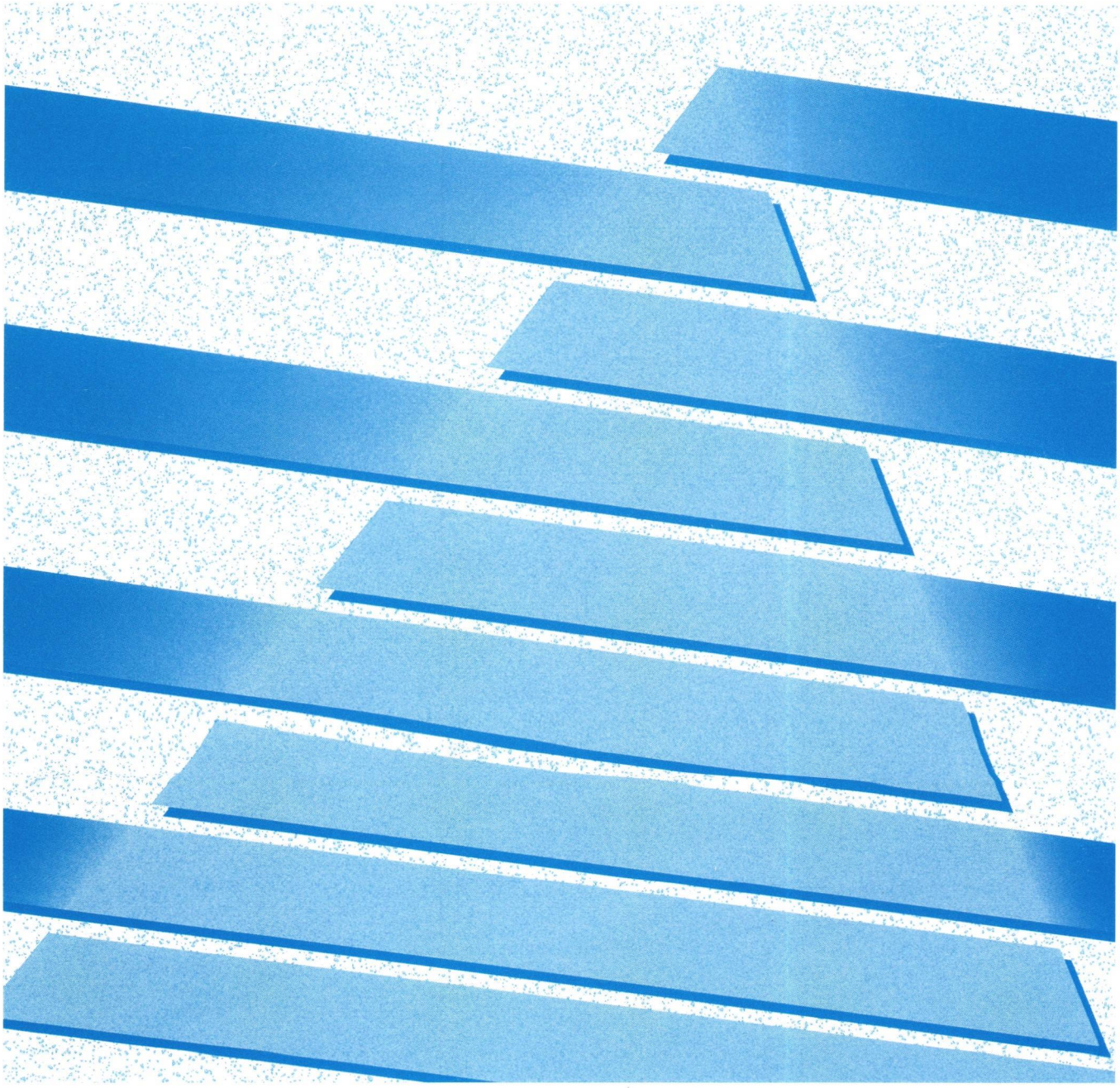
ALLEN-BRADLEY

ControlView™ GE Driver

(Cat. No. 6190-GED)

User Manual



Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards.

The illustrations, charts, sample programs and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, the Allen-Bradley Company, Inc. does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Allen-Bradley Publication SGI-1.1, "Safety Guidelines for the Application, Installation and Maintenance of Solid State Control" (available from your local Allen-Bradley office) describes some important differences between solid-state equipment and electromechanical devices which should be taken into consideration when applying products such as those described in this publication.

Throughout this manual we use notes to make you aware of safety considerations:

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage or economic loss.

Attentions help you:

- identify a hazard
- avoid the hazard
- recognize the consequences

Important: Identifies information that is especially important for successful application and understanding of the product.

Preface

How To Use This Manual

This manual describes the features and capabilities of the GE Driver option, a component of the ControlView™ system. The GE Driver software allows ControlView to communicate with GE Fanuc programmable controllers and Genius™ Blocks over the Genius I/O Bus Network.

The manual describes:

- configuring the GE PCIM hardware
- using the ControlView commands that are specific to the GE Driver

Conventions Used in This Manual

This manual follows the same print conventions as the *ControlView Core User Manual.*

Audience

The GE Driver software is a part of ControlView, therefore you should be familiar with ControlView and have the *ControlView Core User Manual* available for reference. A complete list of related publications is contained in that manual.

You should also be familiar with the Genius I/O Bus network, the GE Fanuc Series 90™-70, Series Five™ and Series Six™ programmable controllers, and Datagram communications.

Related GE Fanuc Publications

▪ Series 90-70 User's Manual	GFK-0262
▪ Series 90-70 Genius Bus Controller User's Manual	GFK-0398
▪ Series Five User's Manual	GFK-0122
▪ Series Five Genius Bus Controller User's Manual	GFK-0248A
▪ Series Six User's Manual	GFK-96602

- Series Six Genius Bus Controller User's Manual GFK-0171B
- Genius I/O User's Manual GEK-90486
- Genius I/O System and Communications GEK-90486D-1
- Genius I/O PCIM User's Manual GFK-0074

The above manuals are available through:

GE Fanuc Automation North America, Inc.
P.O. Box 8106
Charlottesville, VA 22906

or through your local GE Fanuc supplier.

Table of Contents

Introduction

Chapter 1

Configuring the GE Driver

Chapter 2

Using the GE Driver

Chapter 3

C-Toolkit Functions for the GE Driver

Appendix A

Chapter 1

Introduction

This manual outlines the features of release 3.0 of the GE Driver, describes the hardware and software required, and shows you how to use the GE Driver. You'll find instructions for:

- configuring the GE PCIM (PC Interface Module), referred to as the Genius Bus Adaptor card
- installing the Genius Bus Adaptor card
- configuring ControlView

Main Features of the GE Driver

With the GE Driver installed, ControlView can communicate with GE Fanuc Series 90-70, Series Five and Series Six programmable controllers on a Genius I/O Bus network.

ControlView supports two communication networks at a given time. With only the GE Driver option, ControlView will support two Genius networks. With another driver, ControlView will support one Genius network and one other communications network.

Hardware and Software Requirements

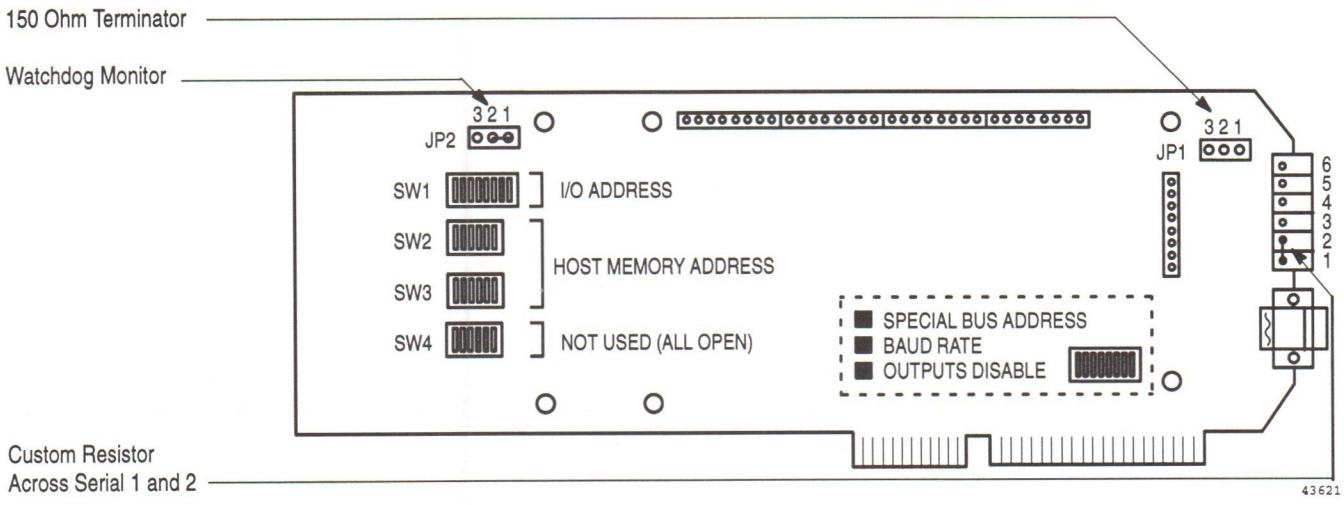
The GE Driver requires:

- the installation of one Genius Bus Adaptor card for each Genius network
- one or more GE Fanuc Series 90-70, Series Five or Series Six programmable controllers

Configuring the PCIM Adaptor Card

The Genius Bus Adaptor card is a full-sized expansion card with a daughterboard. Before installing the Genius Bus Adaptor card in your computer, you must set the card's jumpers and dip switches.

Figure 1.1
The Genius Bus Adaptor Card



Jumper Settings

If the Genius Bus Adaptor card is at the end of the Genius serial bus, you must install a terminating resistor across serial 1 and serial 2. The value of the resistor depends on the type of cable used for the Genius Bus. Refer to Chapter 2 of the *Genius I/O User's Manual* for cable types and resistor values.

Table 1.A
Jumper Settings

Jumper	Setting	Results
JP1	1–2 2–3	150Ω across serial 1 and 2 Any other resistance: connect custom resistor across serial 1 and 2, as shown in Figure 1.1.
JP2	1–2	Watchdog disabled

DIP Switch Settings

Interface Card

Set the four dip switches on the interface card:

- SW1 sets the base address, the first of four contiguous I/O ports in the processor I/O space. In most applications, 3E0(hex) is a good choice.

Example of switch settings for SW1:

Base I/O Port	SW1							
	1	2	3	4	5	6	7	8
3E0 (hex)	C	C	C	O	O	O	O	O
3E4 (hex)	O	C	C	O	O	O	O	O
3E8 (hex)	C	O	C	O	O	O	O	O
3EC (hex)	O	O	C	O	O	O	O	O

O = Open = 1 C = Closed = 0

- SW2 and SW3 set the Host memory address, also known as the Shared Ram Interface (SRI). This is 16k of memory shared by the PCIM and the host computer; no other device or program can use this area.

Example of switch settings for SW2 AND SW3:

Address	SW3						SW2					
	1	2	3	4	5	6	1	2	3	4	5	6
A000	O	X	X	X	X	X	O	C	O	C	C	C
A400	O	X	X	X	X	X	O	C	O	C	C	O
A800	O	X	X	X	X	X	O	C	O	C	O	C
AC00	O	X	X	X	X	X	O	C	O	C	O	O
B400	O	X	X	X	X	X	O	C	O	O	C	O
B800	O	X	X	X	X	X	O	C	O	O	O	C
C000	O	X	X	X	X	X	O	O	C	C	C	C
C400	O	X	X	X	X	X	O	O	C	C	C	O
C800	O	X	X	X	X	X	O	O	C	C	O	C
CC00	O	X	X	X	X	X	O	O	C	C	O	O
D400	O	X	X	X	X	X	O	O	C	O	C	O
D800	O	X	X	X	X	X	O	O	C	O	O	C

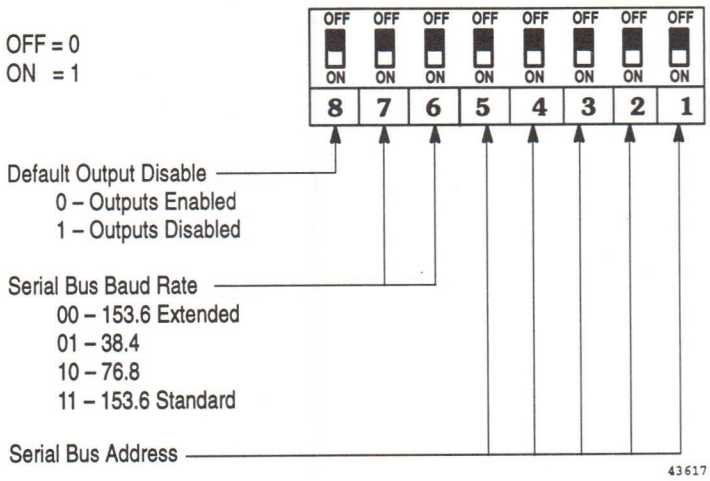
O = Open = 1 C = Closed = 0 X = Don't Care

- SW4 sets the hardware interrupt lines. Set SW4 in the OPEN position (disabling hardware interrupts).

Daughterboard

The daughterboard has a single bank of 8 dip switches:

Figure 1.2
Daughterboard DIP Switches



- Switches 1–5 set the Serial Bus address, a decimal number from 1 to 31
- Switches 6 & 7 set the Serial Baud rate:

Table 1.B
DIP Switch Settings for Serial Baud Rate

Switch setting	Baud rate
00	153.6 Extended (8 bit skip time)
01	38.4 (8 bit skip time)
10	76.8 (8 bit skip time)
11	153.6 Standard (4 bit skip time)

- Switch 8 enables/disables the Default Output:
  - 0 = Outputs Enabled
  - 1 = Outputs Disabled – recommended setting

Installing the Genius Bus Adaptor Card

The Genius Bus Adaptor card requires two expansion slots on the ISA bus.

Tools Required for Installation

To install or remove the Genius Bus Adaptor card, you will need the following tools:

- medium-sized flat bladed screwdriver
- medium-sized Phillips screwdriver

Installation Procedure

Follow these steps to install the card:

1. Record the card's serial number for future reference.
2. Turn the computer OFF and disconnect the power source.

ATTENTION: Hazardous voltages are present in the computer's internal power supply. Turn off the computer and unplug the power cord prior to ANY connection, installation or inspection.

---

3. Disconnect all peripherals and move external devices away from the computer.
4. Remove the computer's cover.
5. Choose an unused slot with space to its right. Remove the cover plate for the slot and keep the screw.
6. Ensure that the jumpers are correctly installed on the card (see Figure 1.1 and Table 1.A).
7. Install the Genius Bus Adaptor card in an open slot with another open slot to the right. For optimal heat dissipation, install the card in a slot as far to the right as possible (closest to the center of the computer).

ATTENTION: The Genius Bus Adaptor card can be damaged by static electricity. To prevent damage to the card, hold the card (still in its antistatic bag) in one hand, and momentarily touch a metal part of the computer with the other hand. Do not touch the card directly!

8. Secure the Genius Bus Adaptor card to the computer chassis with the screw from the cover plate (in Step 5).

9. Reinstall the computer cover.

10. Connect the Genius Communications Cable to the Genius Bus Adaptor card.

ATTENTION: An unshielded plug or cable may cause radio frequency interference. The cable must be properly shielded and properly attached to the Genius Bus Adaptor card.

11. Reconnect peripheral devices and interface cables to the computer.

12. Reconnect the power and restart your computer.

For more information on configuring and installing the Genius Bus Adaptor card, refer to chapter 3 of the *Genius I/O PCIM User's Manual.*

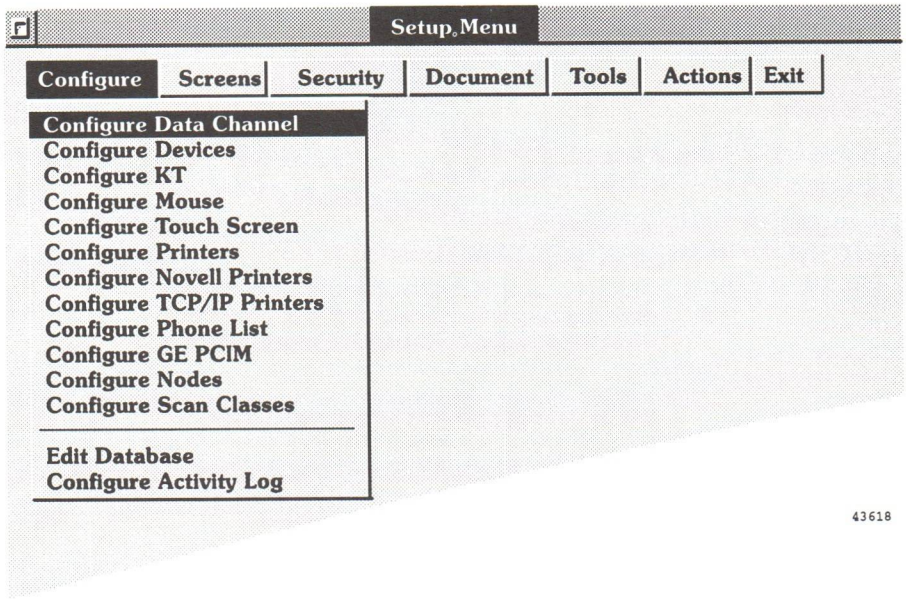
Chapter 2

Configuring the GE Driver

To configure the GE Driver, use the ControlView Setup Menu. When you start up ControlView, the first menu to appear is the Setup Menu.

Important: The Setup Menu appears in the default ControlView system. If you have modified ControlView so that the Setup Menu does not open, open the configuration windows from the command line.

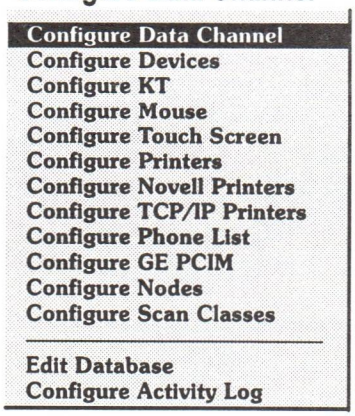
Figure 2.1
The Setup Menu



In the Configure menu, the items which involve GE Driver configuration are:

- Configure Data Channel
- Configure GE PCIM
- Configure Nodes
- Configure Scan Classes
- Edit Database

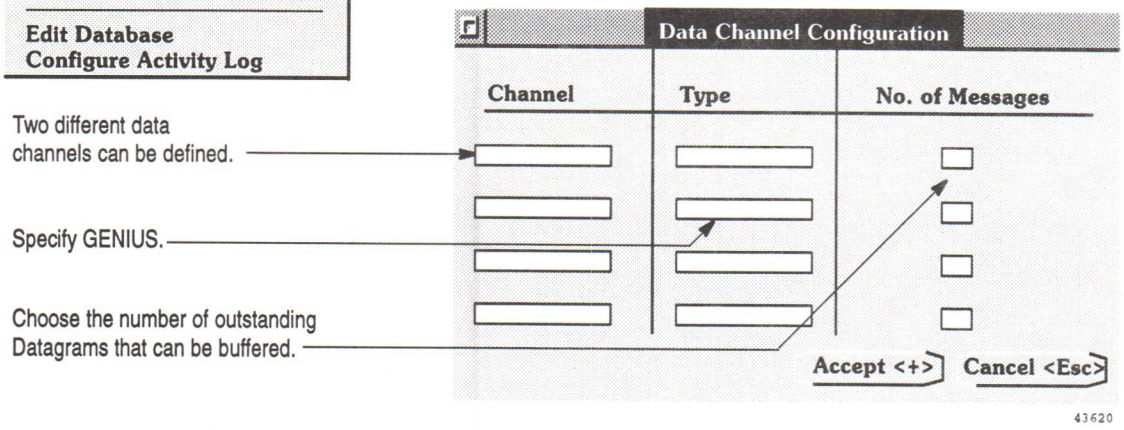
Configure Data Channel



To set up the data channel(s) you'll be using, follow these steps:

1. Choose *Configure Data Channel*. The Data Channel Configuration window will open.

Figure 2.2
Data Channel Configuration Window



2. Fill in these fields for each Genius Bus network (data channel) you are configuring. ControlView can support two Genius networks, provided you have installed two Genius Bus Adaptor cards.

   - Channel

     When the Channel field is highlighted, press **Enter**. A list of choices pops up. Choose *Highway1* or *Highway2*.

   - Type

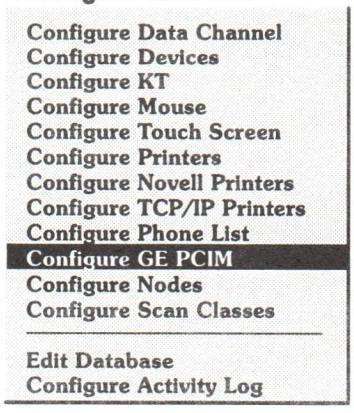
     Choose *GENIUS* from the list.

   - Number of Messages

     Choose a number from 1 to 4 from the list. This field sets the number of unanswered Datagrams ControlView will send to a node before waiting for an answer.

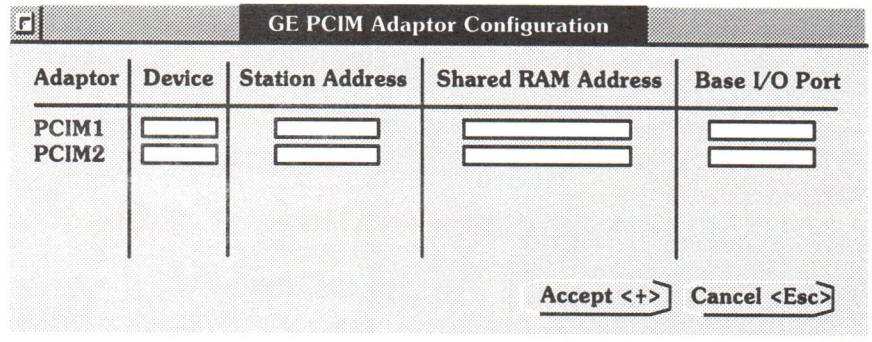
3. When you're finished, click on the *Accept* button to save the information and return to the Configure menu.

Configure GE PCIM



Choose *Configure GE PCIM* under Configure in the Setup menu. The GE PCIM Configuration window will open. Fill in the fields as follows:

Figure 2.3
The GE PCIM Adaptor Configuration Window



- Device

  From the list that pops up, choose the same highway (*Highway1* or *Highway2)* that you configured for GENIUS in the *Type* field of the Data Channel Configuration window.

- Station Address

  Type in the decimal number (*1–31*) to match the setting of DIP switches 1–5 on the PCIM daughterboard.

- Shared RAM Address

  Type in the address that matches the setting of DIP switches SW2 and SW3 on the PCIM interface board. This is the starting address of the Host memory or Shared RAM Interface.

- Base I/O port

  Type in the address that matches the setting of SW1 of the PCIM interface board; commonly 3E0 (hex).

For more details on settings, see your *Genius I/O PCIM User's Manual.*

Restart ControlView to Initialize the Devices

For the changes you've made to take effect, choose *Quit to DOS* in the Exit menu, and then restart ControlView.

Configure Nodes

Use the Node Configuration window to define each GE Fanuc programmable controller that ControlView will communicate with.

Configure Data Channel
Configure Devices
Configure KT
Configure Mouse
Configure Touch Screen
Configure Printers
Configure Novell Printers
Configure TCP/IP Printers
Configure Phone List
Configure GE PCIM
Configure Nodes
Configure Scan Classes

Edit Database
Configure Activity Log

In the Node Configuration window, each programmable controller that ControlView will communicate with is given a name and an associated configuration. The programmable controller is then referred to by that name. This way the attributes such as programmable controller type and station number, which network it's on, etc. needn't be repeated; the node name carries with it all that information.

Up to 32 physical devices can be connected to a Genius network. Since ControlView allows you to assign more than one node name to the same node, you use the 32 devices to create a great number of nodes.

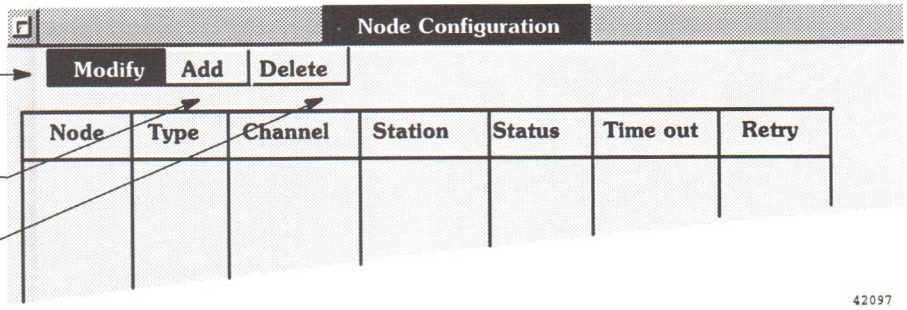
1. Choose *Configure Nodes* from the Configure menu, and the Node Configuration window appears.

Figure 2.4
Node Configuration Window

Select a node and choose *Modify*. You can then change any of the fields. Press + to save the changes, or **Esc** to abandon them.

Choose *Add* and an empty input window pops up. Fill in the fields and press + to save the information, or **Esc** to cancel the addition.

Select a node and choose *Delete*. Choose Accept, to confirm the deletion or **Esc** to cancel the deletion.



2. Fill in the fields in the Node Configuration window.

   - Node

     A name of your choice (up to 8 characters), to represent a programmable controller on the network.

   - Type

     Choose *GE90-70*, *GEFive* or *GESix*.

   - Channel

     Choose *HIGHWAY1* or *HIGHWAY2*, whichever you chose in Data Channel Configuration, and whichever the programmable controller is connected to.

   - Station

The physical station number of the programmable controller on the Genius network. Enter a decimal number between 1 and 31. 0 is reserved for the Hand Held Monitor.

- Status

  Choose *Enabled* or *Disabled.* Normally nodes are enabled, allowing information to be collected. However, during setup or trouble shooting it may be necessary to disable a node to avoid communications timeouts or invalid data.

  When a node is disabled, tag read/writes refer to the Current Value Database (CVD), not the programmable controller(s). Tag values can still be altered in the CVD with the Tag menu options, Set a Tag Value and Ramp Tag Value, and with the SET and RAMP commands.

  Important: Accessing direct programmable controller addresses when a node is disabled produces errors.

- Timeout

  The time (0 to 65535 seconds) to wait before trying again, if there's a communications problem. The default for GE Fanuc programmable controllers is 1.

- Retries

  The number of times to attempt to establish communications before giving up and reporting a communications error. The default setting for GE Fanuc programmable controllers is 3.

3. Unload and reload the database to see your changes. If a database was loaded when you started editing the nodes, the changes will not take effect until you unload the database and load it again.

Configure Scan Classes

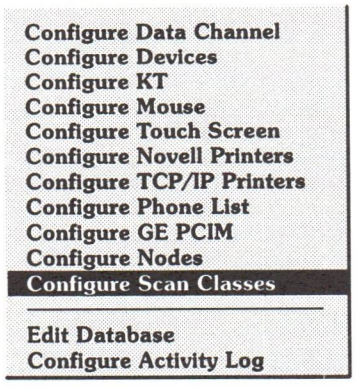
To "scan" is to read the value at a programmable controller address. The scan *period* is how often the address is scanned. ControlView has eleven scan classes, each of which has a foreground and background scanning period.

The numbers you enter in the Foreground and Background Period fields of the Scan Class Configuration window set how many seconds elapse between scans. A setting of 1 scans programmable controller addresses once a second. The longest period is 99,999 seconds (just over 28 hours). 0 seconds means no wait time—the scan will be performed as often as the highway can provide information. Use the 0 period with caution, since the performance of background tasks (such as Alarming) can be compromised when the foreground task tries to scan at top speed.

When defining the scan periods, keep in mind that any tag value in the database is only as accurate as the last scan. If scanning is too slow, the information isn't current. If all scanners are set to high speed scan, network traffic may increase to the point where system performance suffers. The concept of scan classes allows you to optimize your system's performance – to provide high-speed scanning where it's required, and to save on system resources by using lower scan period wherever it's acceptable.

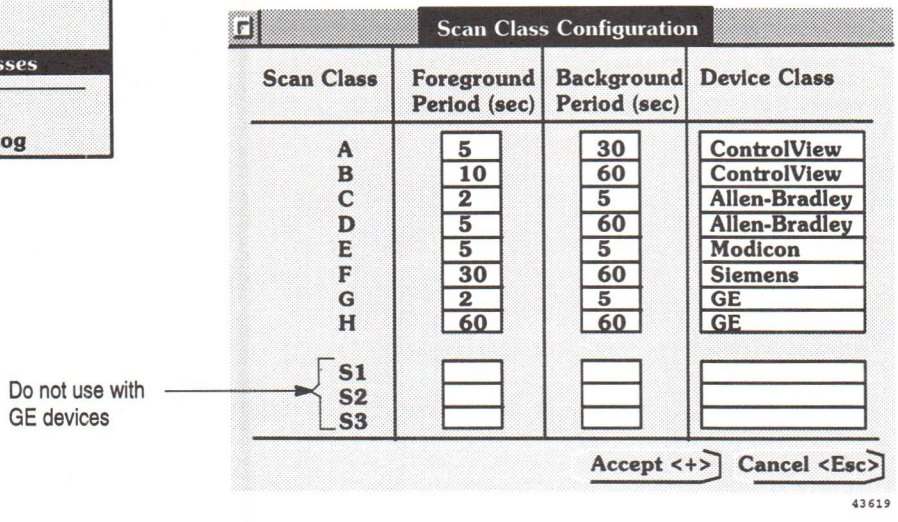
The scan classes have letter names, A through H (for digital and analog tags) and S1 through S3 (for string tags). There is no priority in these names – any of them can have any scan period. You must configure at least one scan class to apply to your Genius network.

Important: Do not assign scan classes S1, S2 and S3 to GE devices. The GE Driver does not support string tags.



1. Choose *Configure Scan Classes* from the Configuration menu and the Scan Class Configuration window appears.

Figure 2.5
The Scan Class Configuration Window



2. Move the highlight to the letter name you want to configure.

3. Fill in the following fields for GE Fanuc controllers:

   - Foreground Period

     Choose a rate for foreground applications such as Mouse GRAFIX™ screens or Trend graphs.

   - Background Period

     Choose a rate for background applications such as Alarming, Event Detector, and Derived Tags.

   - Device Class

     Choose *GE* from the list.

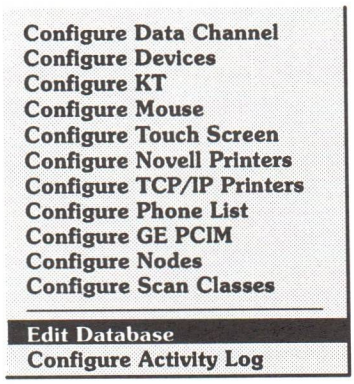
     Important: You must use separate scan classes for different highway types, i.e., a point in a GE Fanuc programmable controller cannot belong to an Allen-Bradley scan class.

4. Choose *Accept* to save the information in the Scan Class Configuration window to disk, and to return to the Setup Menu.

Unload and Reload the Database to See Your Changes

The Scan classes that apply to a database are loaded with the database. If a database was loaded when you changed the scan classes, you will not see the changes until you unload the database and load it again.

The Database and Addressing



To add a point in the database, choose *Edit Database* from the Configure menu and choose the database; then choose *Add* from the Configure Database window menu and specify your tag type: digital, analog or structure.

Enter a properly formatted address in the Configure Digital point window (Figure 2.6), the Configure Analog Point window (Figure 2.7), or the Configure Structure window (Figure 2.8). (For structures, this is a base address. Analog and digital structure members are configured as an offset from this base.)

Figure 2.6
Configure Digital Point Window

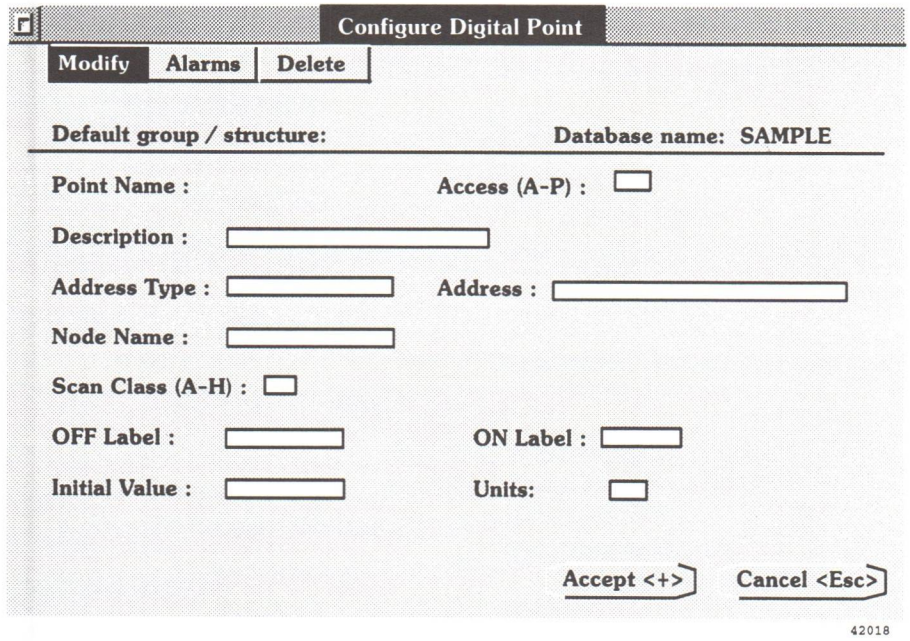


Figure 2.7
Configure Analog Point Window

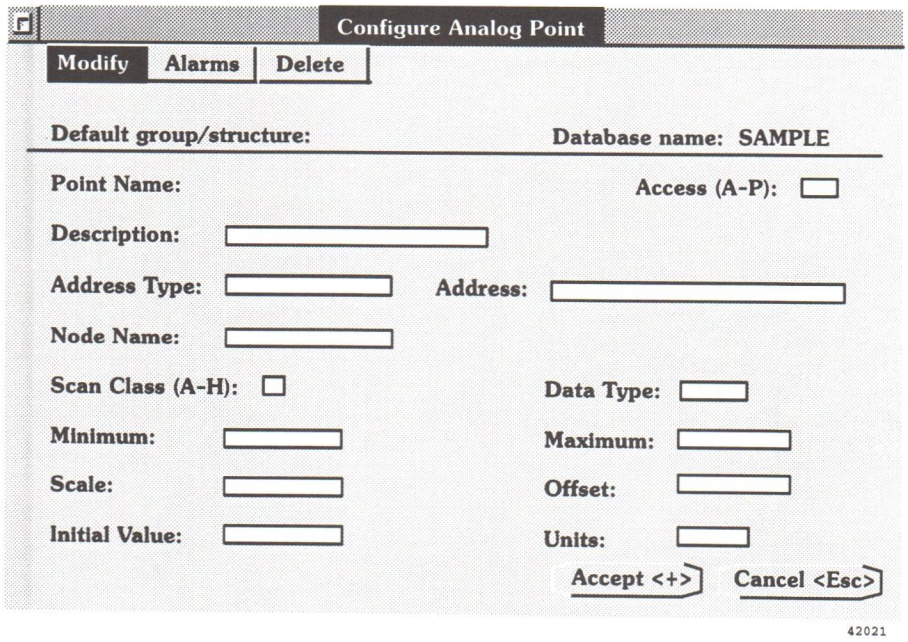
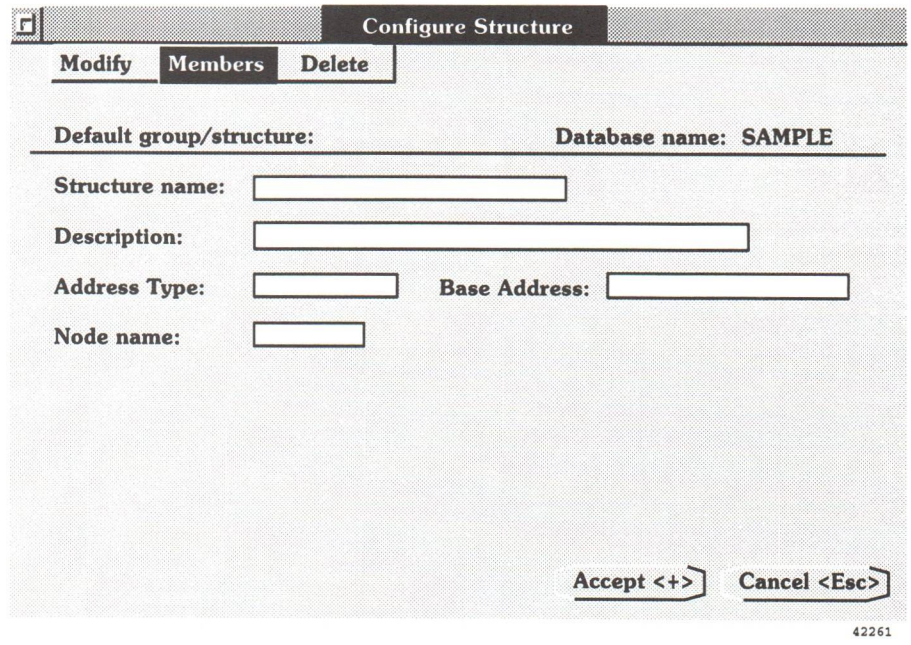


Figure 2.8
Configure Structure Window



Only the Address Type and (Base) Address fields are affected by the use of a Genius network. Refer to Chapter 3 of the *ControlView Core User Manual* for more information on the database editor and how to configure points.

- Address Type

  Choose *GE90-70, GESix* or *GEFive* for the GE Fanuc family of programmable controllers. For local tags, choose *None*.

- Address/Base Address

  Specify the exact memory location in the programmable controller.

GE Driver Address Syntax

Addressing for the GE Fanuc programmable controllers is as follows:

Series 90-70 Programmable Controllers

%MMwwwww *%MM* = Memory Type — Analog points only
- %R = Register Memory (16-bit words)
- %AI = Analog Input Memory (16-bit words)
- %AQ = Analog Output Memory (16-bit words)

wwwww = word address

%MMbbbb *%MM* = Memory Type — Digital points only
- %I = Discrete Input Memory
- %Q = Discrete Output Memory
- %T = Discrete Temporary Memory
- %M = Discrete Momentary Memory
- %SA = Discrete System Memory Group "A"
- %SB = Discrete System Memory Group "B"
- %SC = Discrete System Memory Group "C"
- %S = Discrete System Memory
- %G = Discrete Genius Automatic Global Data Table

bbbb = bit address

Series Six Programmable Controllers

Register Memory syntax: Analog points only

Rwwwww R = Register memory (16 bit words)

wwwww = word address: 1–8192 for 8k systems; 1–16384 for 16k systems

Discrete Real Input and Output syntax for channels 1–7 and 9–F: Digital points only

Mx+bbbb *M* = Memory Type
- I = Input
- O = Output

x = channel number: 1–7 or 9–F

bbbb = bit address: input or output point number: 1–1024

Discrete Real Input and Output syntax for channel 0: Digital points only

Mbbbb *M* = Memory Type
- I = Input
- O = Output

bbbb = bit address: input or output point number: 1–1024

Discrete Real Input and Output syntax for channel 8: Digital points only

AMbbbb

M = Memory Type
- I = Input
- O = Output

bbbb = bit address: input or output point number: 1–1024

Discrete Internal Input and Output syntax: Digital points only

Mx-bbbb

M = Memory Type
- I = Input
- O = Output

x = channel number: 0–F

bbbb = bit address: input or output point number: 1–1024

Series Five Programmable Controllers

Register Memory syntax: Analog points only

Rwwwww

R = Register memory type

wwwww = word address: 1–16384

Discrete Local I/O syntax: Digital points only

Mbbbb

M = Discrete Local memory type
- I = Discrete Local Inputs
- O = Discrete Local Outputs

bbbb = bit address: 1–1024

Discrete Remote and Internal I/O: Digital Points only

MC±bbbb

M = Discrete memory type
- I = Inputs
- O = Outputs

C = Channel number: 1 or 2

+ = Remote

– = Internal

bbbb = bit address: 1–1024

Note: I2– is not a valid memory type.

Structure Tags: Base and Offset Addressing

To define a structure tag, choose *Structure* as the Tag Type in the Add Tag window, then choose *Accept*. The Configure Structure window (see Figure 2.8) appears.

In this window, specify:

- Address Type

  Choose *GE90-70, GESix* or *GEFive*; for local tags, choose *None*

- Base Address

  Enter the address that points to the beginning of the structure. The base address follows the addressing conventions used by a conventional tag.

Structure members are digital or analog tags, defined in the Structure Digital Point and Structure Analog Point windows. These windows have a field called *Address offset*. Identify each member in the structure by specifying its offset from the base address defined in the Configure Structure window. ControlView calculates the final programmable controller address for each structure member by adding the base and offset together.

Important: Do not mix digital and analog members in the same structure.

The offset is in words for analog tags, in bits for digital tags. Offsets are in decimal for GE Fanuc controllers.

Example: Valid Base & Offset Addresses for Different Programmable Controllers

Table 2.A
Sample Base and Offset Addresses

Address Type	Base Address	Offset	Actual Address
GE90-70	%R00001	10	%R00011
	%I00001	10	%I00011
GESix	R00001	10	R00011
	I1+0001	10	I1+0011
GEFive	R00001	10	R00011
	I1+0001	10	I1+0011

String Tags

The GE Driver cannot read or write string tags.

Chapter 3

Using the GE Driver

This chapter tells you how to verify that the GE Driver is properly installed and configured by using it to read and write to the programmable controller.

Diagnostics

To confirm proper configuration of the GE Driver, under Tags, in the Action menu:

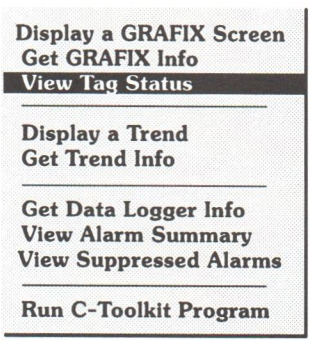
1. Choose *View Tag Status* to verify that ControlView can read from a programmable controller.
2. Choose *Set a Tag Value* to verify that ControlView can write to a programmable controller.
3. Choose *View Tag Status* again to verify that the tag value has, in fact, changed.

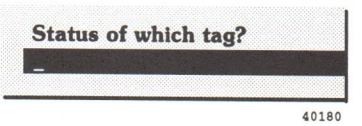
Important: The Data Channel, Device and Node configuration must be completed before these operations will work correctly.

Reading from the Programmable Controller

To read a value from a programmable controller:

1. Load a database.
2. Choose *View Tag Status* under Tags in the Actions menu. A window appears.





40180

3. Type in a tag name, or use wild card characters to refer to several tag names .

If a single tag name is given for location, database information (such as description, node, and address) will be displayed, along with the tag name and its current value.

If several tags are specified, a list of the tag names and values will be displayed. Up to seven tags can be displayed in the window; the rest can be seen by paging though the list (with **PgUp** and **PgDn**). The list can contain a maximum of 100 values.

Instead of specifying a tag name (with or without wild cards), you can specify a physical programmable address. The format for specifying a node name and physical address is:

$*nodename::address* (There are no spaces in the string)

dollar sign ($)	distinguishes a node name from a tag name.
nodename	is specified in the Node table
address	is the physical programmable controller address syntax

A valid GE 90-70 programmable controller address, for example, looks like this:

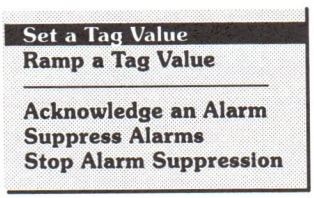
```
$fred::%I00001
```

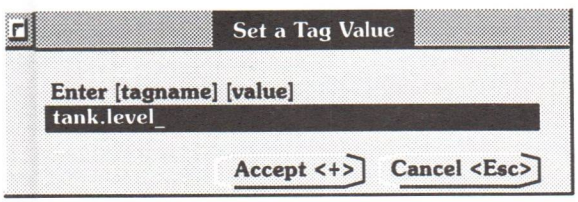
If you can call up the Tag Status window displaying information on the tags in your database, your GE Driver is reading the programmable controller correctly.

Writing to the Programmable Controller

Set a Tag Value lets you write a value directly to a programmable controller address.

1. Load a database.
2. Choose *Set a Tag Value* and a data-entry window will open.





3. Type in the name of the tag and the value to write, in this form:

<tagname> <value>

In this expression, *<tagname>* is either:

- a valid tag name, such as:

  `tank.level`

  OR

- a node name and physical address in this format:

  $*nodename::address* (There are no spaces in the string)

  dollar sign ($) distinguishes a node name from a tag name.

  nodename is specified in the Node table

  address is the physical programmable controller address syntax

  A valid programmable controller address looks like this:

  `$fred::%I00001`

And *<value>* is one of the following:

For analog tags:

- A numeric value within the range specified by the tag's Minimum and Maximum values in the tag database.

  If the value specified is outside the range between Minimum and Maximum, the write will not be performed, and an error message will be displayed

- A percentage of the total range between Minimum and Maximum. The formula is:

  value = Min + (percentage/100 * (Max – Min))

  Note: For direct addressing, the value must be within the valid data range of the physical address being written to.

For digital tags:

- The numeric value zero (0) or one (1)

- The tag's ON or OFF label as specified in the database. Setting a digital tag to its ON label will write the value 1 to the programmable controller; setting it to its OFF label will write the value "0"

Only numeric values can be used when writing directly to a programmable controller location, since Minimum, Maximum, and ON/OFF labels are not recognized by the controller.

Examples: Set Tag Value

The following examples are responses to the prompt:

```
Enter [tagname] [value]
```

`tank.level 50%`
if Minimum = –100 and Maximum = 900, the command would write the value 400 to the programmable controller, according to the formula above:

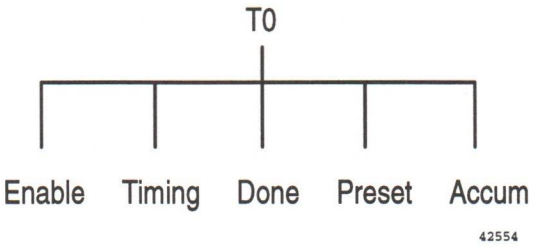
value = –100 + (50/100 * (900 – – 100))
= –100 + (1/2 * 1000)
= –100 + 500
= 400

`valve.23 open`
uses the ON label for this digital tag to write a "1" to the programmable controller.

Important: You can set the tag value of any structure member in a structure tag, but not of the structure tag itself.

Example: Structure Tags and the SET Command

The database has a structure called T0 which represents a timer. This database structure looks like this:



This command is not allowed:

```
SET T0 1
```

and would result in an error, while this command would have the desired result:

```
SET T0.PRESET 1
```

ATTENTION: If a node has been disabled, the SET command will change the Current Value Database value, but not the programmable controller's value itself.

Check the operation of the GE Driver by setting several tags and then viewing their status. If you can change the values in your programmable controller and read the new data back, your GE Driver is functioning properly.

The Communication Status Display

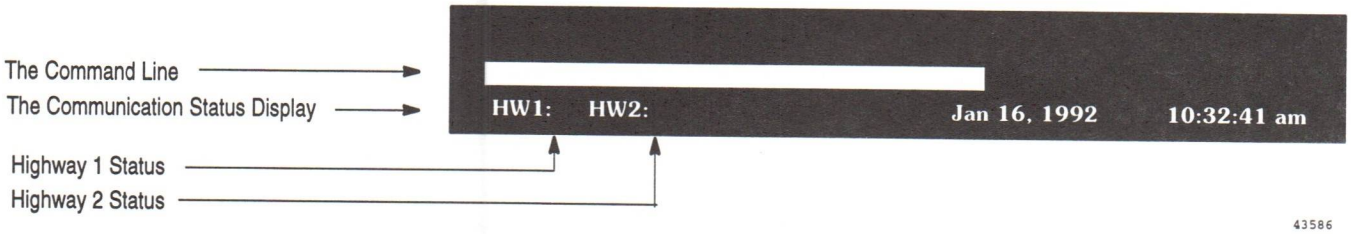
Choose *Communication Status* under Tools in the Actions menu to bring up the Communication Status Display. A list of three choices will appear:

Choose *station*; this configures the Communication Status display to show the station number of the programmable controller that ControlView is currently communicating with; under normal conditions, a green display will continuously update with current information.

Do not choose *messages* unless you want to configure the Communication Status display to show the number of buffered messages (Datagrams) waiting to be transmitted.

The Communication Status Display will appear at the bottom of the screen.

Figure 3.1
The Communication Status Display



If a communication error occurs, the display will change to reverse-video red. When the error has cleared, the display will change to yellow to indicate that an error had occurred. These are default colors; you can change them by choosing *Set Up COMSTATUS Display* under Tools in the Setup menu.

Choose *reset* to acknowledge the error and restore the indicators for both channels to green after an error has been detected. If you are working in an operator's screen, a Mouse GRAFIX display, for example, you can reset the indicators more readily by calling up the command line (press **Alt-C**) and entering the command:

COMSTATUS RESET *press* ***Enter***

You can use the COMSTATUS command to control the Communications Status display. Refer to Appendix A, *ControlView Commands* in the *ControlView Core User Manual* for more detail.

The display format for the COMSTATUS window is:

`HW1:`*`nnn`* `HW2:`*`nnn`*

`HW1` indicates HIGHWAY1 (as defined in the Device Configuration window) information

`HW2` indicates HIGHWAY2 information (if defined)

`nnn` is the actual information, either the current station number, or the number of buffered message transactions

Important: Genius Bus information is displayed in decimal, whereas Allen-Bradley programmable controller information is displayed in octal.

Logging Communication Errors

You can choose to log all tag reads, tag writes, and communication activity, including communication errors, to a printer or disk file.

To log all communications activity, make sure to specify that you want Communications, Tag Writes, and Tag Reads when configuring the activity log. See Chapter 3, *The Setup Menu*, in the *ControlView Core User Manual* for complete details.

Appendix A

C-Toolkit Functions for the GE Driver

Overview

The GE Fanuc Genius I/O Bus network can be used for communications between programmable controllers, computers, and Genius I/O blocks. There are three types of communications possible over a Genius Bus:

- Datagrams
- I/O Service
- Global Data

The ControlView GE Driver software uses Datagram communications to provide a configurable interface between ControlView and GE Fanuc programmable controllers, as described in detail earlier in this manual.

The configurable support for Series 90-70, Series Six, and Series Five programmable controllers will probably satisfy the application requirements of most users who want to use GE Fanuc equipment with ControlView.

However, there may be some users who need to establish a communications link between a ControlView system and some other type of GE Fanuc equipment (such as one of the Genius I/O blocks). You can create this type of communication link by writing a ControlView C-Toolkit program that uses the GE Driver C-Toolkit library distributed with the GE Driver software.

This appendix describes how to use the GE Driver C-Toolkit library. You should already be familiar with the following subjects:

- writing a C-Toolkit program using the ControlView C-Toolkit option
- operation of the GE Fanuc Genius I/O Bus network
- operation of any connected GE Fanuc equipment

The examples that follow are for the Microsoft® C compiler, version 6.0.

Features of the GE Driver C-Toolkit

A ControlView C-Toolkit program can use the GE Driver library to:

- format and send a datagram to any type of Genius Bus device that supports datagram communications. The GE Driver supports any type of datagram that is currently supported by the Genius Bus. If the selected datagram type should expect a reply datagram, the GE Driver will compare incoming datagrams to match the reply datagram to the request datagram.
- control device I/O, either by bit, by word, or by device (up to 64 words). To use this feature, the Genius Bus device number (station number) on the PCIM should be set to 31.
- receive Global Data that is being transmitted by other Genius Bus devices. This can be accomplished by reading the device input table for the selected device.

For more information on using these capabilities of the Genius I/O Bus network, you should consult the "Related GE Fanuc Publications" listed in the preface of this manual.

Using the GE Driver C-Toolkit

When the GE Driver software is installed with the procedure described earlier in this manual, the following files are placed in the ControlView directory structure:

```
\ACCESS\GED\INCLUDE\CTKGED.H
\ACCESS\GED\LIBRARY\CTKGEDMS.LIB
\ACCESS\GED\LIBRARY\CTKGEDMM.LIB
```

The files listed above are distributed with the GE Driver option, rather than the C-Toolkit option, since many C-Toolkit users will have no need of the GE Driver functions. The file CTKGED.H is the header file for the library, and should be referenced with an #include directive inside any C source file that contains calls to the GE Driver functions. CTKGEDMS.LIB is the object code library to be used with small memory model programs. CTKGEDMM.LIB is the medium memory model version of the library.

You should modify any make files or batch files that you are using to build C-Toolkit applications to reflect the fact that you are using the GE Driver files. In particular, you must identify either CTKGEDMS or CTKGEDMM on the linker command line.

Example: Linker Command Line Parameters

```
cl/AS/c test1.c
link/NOE mcsctk+test1, test1.tsk,, mcsctk+slibce+ctkgedms;
```

The LIB environment variable should be set with the path to the GE Driver library files, in addition to the path to the C-Toolkit option library files and your compiler's library files. The INCLUDE environment variable should be set with the path to the GE Driver include file, as well as the path to the C-Toolkit option include files and your compiler's include files. This is typically accomplished by adding lines to the AUTOEXEC.BAT file or some other batch file.

Example: Setting INCLUDE and LIB Environment Variables

```
set INCLUDE=c:\c600\include;c:\access\ctk\include;c:\access\ged\include;
set LIB=c:\c600\lib;c:\access\ctk\library;c:\access\ged\library;
```

To communicate through a PCIM adaptor using the ControlView GE Driver C-Toolkit, you must first configure the PCIM and a Data Channel, using the techniques described earlier in this manual. It is not necessary to configure Nodes, Scan Classes, or database tags, since the functions provided in the GE Driver C-Toolkit are "low-level" functions that operate with raw data.

Return Code Reference

GED_SUCCESS = the function call was successfully completed, and the reply or input data (if applicable) has been written to the user's buffer.

GED_SYSTEM_ERROR = an error of undeterminable cause was encountered during the function call.

GED_INVALID_HWY = hwy_number is not a valid highway number (1 or 2).

GED_NO_DRIVER = either your PCIM adaptor is not configured properly, or it was not successfully initialized by the ControlView GE Driver when ControlView was started.

GED_FORMAT_ERROR = either the selected Genius Bus device is not present on the Genius Bus, or there is some problem in processing the parameters that were passed in the function call.

GED_NO_RESPONSE = the remote device is present on the Genius Bus, but there was no reply received before time-out.

Detailed Function Reference

The following section of this appendix contains a detailed description of all functions in the GE Driver C-Toolkit library. The symbolic constants for the return codes are defined in the CTKGED.H file.

ctk_ged_datagram()

ctk_ged_datagram() sends a datagram to another device over the Genius I/O Bus, and in some cases returns a reply datagram from the remote device. The contents of the datagram are completely controlled by the C-Toolkit program, and the GE Driver does not perform any validity checks on the datagram.

C Synopsis

```
#include <ctkged.h>

unsigned int ctk_ged_datagram (unsigned int hwy_number,
                               SEND_MESSAGE* send_message,
                               unsigned char* receive_buffer);

typedef struct
{
   unsigned char destination; /*Destination address for datagram*/
   unsigned char function;    /*Function Code, normally set to
   0x20*/
   unsigned char subfunction; /*Specifies type of datagram */
   unsigned char priority;    /*0 = normal, 1 = high */
   unsigned char length; /* length of "data", max=134 bytes */
   unsigned char data [1134]; /*Datagram to be transmitted */
}SEND_MESSAGE;
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

send_message — a pointer to a structure of type SEND_MESSAGE, which is defined in CTKGED.H

send_message -> destination
the device number of the remote device on the Genius Bus (0 - 31, decimal).

send_message -> function
should be set to DATAGRAM_FUNCTION_CODE.

send_message -> subfunction
a code which defines the type of datagram, taken from the list of subfunction codes in CTKGED.H. This list of codes matches the list found in the *GE Fanuc Genius I/O System and Communications* manual.

send_message -> priority
0 for normal priority, 1 for high priority.

send_message -> length
: the length of the datagram to be sent, which is also the number of relevant bytes in the "data" field which follows.

send_message -> data[]
: the datagram data, which may vary greatly from one type of datagram to another. You should consult the *GE Fanuc Genius I/O System and Communications* manual and the user's manual for the remote device to determine the correct format for the type of datagram you are using.

Output Details

receive_buffer
: a pointer to a buffer of MAX_DATAGRAM_LEN bytes, where the reply datagram should be stored. If receive_buffer is NULL, the function will not return a reply datagram, whether a reply should be expected or not. The following datagram types, from the list in CTKGED.H should cause a reply datagram to be sent from the remote device:

Datagram Transmitted	Expected Reply Datagram
READ_ID_DTG	READ_ID_REPLY_DTG
READ_CONFIG_DTG	READ_CONFIG_REPLY_DTG
READ_DIAGS_DTG	READ_DIAGS_REPLY_DTG
READ_BLOCK_IO_DTG	READ_BLOCK_IO_REPLY_DTG
PULSE_TEST_DTG	PULSE_TEST_COMPLETE_DTG
READ_DEVICE_DTG	READ_DEVICE_REPLY_DTG
READ_DATA_DTG	READ_DATA_REPLY_DTG
READ_MAP_DTG	READ_MAP_REPLY_DTG

If you are using one of the subfunction codes in the list on the left above, you should make sure that you set **receive_buffer** to the address of a buffer of **MAX_DATAGRAM_LEN** bytes. If you are using any other type of datagram, including those in the list on the right above, you should set **receive_buffer** to NULL.

The following list contains the subfunction codes which are not found in the lists above. These datagrams do not cause a reply datagram to be sent from the remote device. If you are using one of these datagram types, you should set receive_buffer to NULL:

WRITE_CONFIG_DTG
ASSIGN_MONITOR_DTG
BEGIN_PACKET_SEQ_DTG
END_PACKET_SEQ_DTG
WRITE_POINT_DTG
REPORT_FAULT_DTG
CLEAR_CKT_FAULT_DTG
CLEAR_ALL_CKT_FAULTS_DTG
SWITCH_BSM_DTG
WRITE_DEVICE_DTG
CONFIG_CHANGE_DTG
WRITE_DATA_DTG
WRITE_MAP_DTG

Return Codes

GED_SUCCESS
GED_INVALID_HWY
GED_FORMAT_ERROR
GED_SYSTEM_ERROR
GED_NO_DRIVER
GED_NO_RESPONSE

Notes

A complete discussion of the usage and contents of the datagram types listed above is beyond the scope of this manual. You should consult the "Related GE Fanuc Publications" listed in the preface of this manual to obtain the appropriate information concerning the equipment you are using.

Example: ctk_ged_datagram()

```
/*
**Read device datagram using ctk_ged_datagram() - reads % R1 from a Series 90-70 PLC
**which has a Genius Bus device number of 2.
*/

#include <stdio.h>
#include <conio.h>
#include <ctk.h>
#include <ctkged.h>
#define  WDW_ORG_T     20
#define  WDW_ORG_Y     200
#define  WDW_W         400
#define  WDW_H         100
#define BACK_COLOR  WHITE
void main {void}
{
SEND_MESSAGE datagram;
  unsigned char Datain [MAX_DATAGRAM_LEN];
  unsigned int status;
  ctk_make_window (WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  /*
  **Set up SEND_MESSAGE structure:
  */
  datagram.destination = 2;      /* Remote device number of Series 9--70 */
  datagram.function = DATAGRAM_FUNCTION_CODE;      /* Always */
  datagram.subfunction = READ_DEVICE_DTG;
   /* Read Device Datagram */
  datagram.priority=0; /* Normal priority message */
  datagram.length=6; /* Length of "data" field which follows */
  datagram.data[0]=0; /* Reserved - set to zero */
  datagram.data[1]=8; /* Memory type: 8=Register memory (90-70 only) */
  datagram.data[2]=0; /* Always zero for 90-70 */
  datagram.data[3]=0; /* Memory Offset, less 1 (LSB) */
  datagram.data[4]=0; /* Memory Offset, less 1 (MSB) */
  datagram.data[5]=1; /* Length, in words, to read from 90-70 */

  /*
  **Send datagram:
  */
  status = ctk_ged_datagram (1, &datagram, Datain);
  if (!status)
  {
    /*
    **The first six bytes returned from the Series 90-70 are the datagram header,
    **so the data we are interesed in is after the first six bytes.
    */
      printf ("R1=%d\n", 256 * Datain [7] + Datain [6];
  }
  else
    printf ("Error in datagram transaction\n");
  while (!kbhit())     /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch ();     /* Remove character from console input buffer */
}
```

ctk_ged_read_status()

ctk_ged_read_status() reads a block of status data from the PCIM adaptor. The status data includes a table of all 32 possible Genius Bus device numbers that indicates which devices are present on the bus.

C Synopsis

```
#include <ctkged.h>

unsigned int ctk_ged_read_status  (unsigned int hwy_number,
                                   GIO_STATUS* gio_status);

typedef struct
{
   unsigned int gio_outtransact; /* number of outstanding
                                    transactions */
}GIO_CFG;

typedef struct
{
   unsigned char pcim_dip_switch; /* DaughterBoard DIP switch
                                     settings */
   unsigned int pcim_imref; /* Global Data reference */
   unsigned char pcim_output_len; /* Global Data output
                                     length */
   unsigned char pcim_input_len; /* Reserved - set to 0 */
   unsigned char pcim_revision; /* pcim firmware revision
                                   number */
   unsigned char pcim_geni_ok; /* PCIM fault code - defined
                                  below */
   unsigned char pcim_active; /* 0 = Hand Held Monitor
                                 Present */
   unsigned int pcim_sberr; /* Serial Bus error roll-over
                               counter */
   unsigned int pcim_scantime; /* Bus scan time in ms */
}GIO_PCIM_STATE;

typedef struct
{
   unsigned char bus_device_model; /* Model number of serial
                                      device */
   unsigned char bus_device_disable; /* 0 = outputs enabled,
                                        1 = disabled */
   unsigned char bus_device_present; /* 0 = not present,
                                        1 = device present */
   unsigned int bus_device_global_ref; /* Global data ref of
                                          broadcasting CPU */
   unsigned char bus_device_input_len; /* Reserved - set to 0 */
   unsigned char bus_device_output_len; /* Global data length */
   unsigned char bus_device_config;
                                   /*1 = input data only device*/
                                   /* 2 = output data only device */
                                /* 3 = input and output data device */
}GIO_BUS_DEVICE:

typedef struct
{
   GIO_BUS_DEVICE   gio_stat_device_table [32];   /* 256 bytes */
   GIO_PCIM_STATE   gio_stat_pcim;                /* 13 bytes */
   unsigned int     gio_stat_sri_seg;             /* 2 bytes */
   GIO_CFG          gio_stat_config;              /* 2 bytes */
   unsigned char    gio_stat_reserved [25];       /* 25 bytes */
} GIO_STATUS;
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

Output Details

gio_status — a pointer to a structure of type GIO_STATUS, to which the status data will be written.

gio_status -> gio_stat_device_table[]
an array of structures of type GIO_BUS_DEVICE. Each of the GIO_BUS_DEVICE structures contains information about one Genius Bus device, as defined in the structure definition above.

gio_status -> gio_stat_pcim
a structure of type GIO_PCIM_STATE which contains information about the PCIM adaptor, as defined in the structure definition above.

gio_status -> gio_stat_sri_seg
the starting segment address of the PCIM shared RAM interface (16kB), which is set with DIP switches and configured in ControlView as described earlier in this manual.

gio_status -> gio_stat_base_port
the base I/O port used by the PCIM adaptor, which is set with DIP switches and configured in ControlView as described earlier in this manual.

gio_status -> gio_stat_config
a structure of type GIO_CFG which contains the number of outstanding transactions that the ControlView GE Driver may have outstanding at once, as configured through the ControlView *Data Channel* configuration.

gio_status -> gio_stat_reserved []
an array of unused bytes which are reserved for future use.

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER

Example: ctk_ged_read_status()

```
/*
**Reads device table and PCIM status using ctk_ged_read_status()
*/
#include <conio.h>
#include <dos.h>
#include <ctk.h>
#include <ctkged.h>
#define WDW_ORG_T 20
#define WDW_ORG_Y 10
#define WDW_W 600
#define WDW_H 320
#define BACK_COLOR WHITE
void main (void)
{
  GIO_STATUS statdata;
  unsigned char* statdata_ptr;
  unsigned int status;
  int i,j;
  ctk_make_window (WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  statdata_ptr = &statdata;
  status = ctk_ged_read_status (1,&statdata);
  if (!status)
  {
    /*
    **Display the status in a rather raw display format:
    */
    for (i=0, j=1; i<275; i++, j++
    {
      if(j==1) printf("%03X",i);
      if(j==9) print("-");
      printf("%02X", statdata_ptr[1]);
      if(j==16)
      {
        printf("\n");
      }
    }
  }
  else
    printf("Error reading status data\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch();    /* Remove character from console input buffer */
}
```

ctk_ged_enable_outputs()

ctk_ged_enable_outputs() enables control outputs on the PCIM adaptor.

C Synopsis

```
#include <ctkged.h>
unsigned int ctk_ged_enable_outputs(unsigned int hwy_number,
                                    unsigned int device_number);
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

device_number — the Genius Bus device number that you wish to enable outputs for. Typically, outputs may simply be enabled for all devices on the bus at once. To accomplish this, set device_number to ALL_DEVICES.

Output Details

No output parameters are returned.

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER
GED_FORMAT_ERROR	GED_NO_RESPONSE

Notes

To use this feature, your PCIM must be configured (with DIP switches) as device 31 on the Genius Bus, as other Genius Bus devices (such as Genius blocks) will only recognize control outputs from device 31. This function overrides the setting of the Enable/Disable Outputs DIP switch on the PCIM daughterboard, which determines whether outputs are enabled upon power-up.

Example: ctk_ged_enable_outputs()

```
/*
**Enables outputs for all devices using ctk_ged_enable_outputs
*/
#include <ctk.h>
#include <ctkged.h>
#define WDW_ORG_T   20
#define WDW_ORG_Y   180
#define WDW_W       400
#define WDW_H       140
#define BACK_COLOR  WHITE
void main (void)
{
  unsigned int status;
  ctk_make_window (WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK COLOR);
  ctk_cursor_off();
  status = ctk_ged_enable_outputs (1, ALL_DEVICES);
  if(!status)
    printf("Outputs enabled\n");
  else
    print("Error enabling outputs\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch();    /* Remove character from console input buffer */
}
```

ctk_ged_disable_outputs()

ctk_ged_disable_outputs() disables control outputs on the PCIM adaptor.

C Synopsis

```
#include <ctkged.h>
unsigned int ctk_ged_disable_outputs (unsigned int
                                      hwy_number,unsigned int
                                      device_number);
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

device_number — the Genius Bus device number to which you wish to disable outputs. Typically, outputs may simply be disabled for all devices on the bus at once. To accomplish this, set device_number to ALL_DEVICES.

Output Details

No output parameters are returned.

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER
GED_FORMAT_ERROR	GED_NO_RESPONSE

Notes

This function overrides the setting of the Enable/Disable Outputs DIP switch on the PCIM daughterboard, which determines whether outputs are enabled upon power-up.

Example: ctk_ged_disable_outputs()

```
/*
**Disables outputs for all devices using ctk_ged_disable_outputs
*/
#include <ctk.h>
#include <ctkged.h>

#define  WDW_ORG_T     20
#define  WDW_ORG_Y     180
#define  WDW_W         400
#define  WDW_H         140
#define BACK_COLOR  WHITE
void main (void)
{
  unsigned int status;
  ctk_make_window (WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  status = ctk_ged_disable_outputs (1, ALL_DEVICES);
  if (!status)
    printf ("Outputs disabled\n");
  else
    printf("Error disabling outputs\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch();    /* Remove character from console input buffer */
}
```

ctk_ged_get_inputs()

ctk_ged_get_inputs() reads the device input table (up to 128 bytes) for one Genius Bus device.

C Synopsis

```
#include <ctkged.h>
unsigned int ctk_ged_get_inputs (unsigned int hwy_number,
                                 unsigned int device_number,
                                 unsigned char* input_buffer);
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

device_number — the Genius Bus device number (0 - 31) from which you wish to read the device input table.

Output Details

input_buffer — a pointer to a buffer of MAX_INPUT_DATA_SIZE bytes, where the device input data will be written.

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER
GED_FORMAT_ERROR	GED_NO_RESPONSE

Notes

This function can be used to receive Global Data transmitted from other bus controllers, since the Global Data from those devices will appear in the appropriate device location within the device input table for the PCIM.

Example: ctk_ged_get_inputs()

```
/*
**Reads device input table from device 1
*/
#include <ctk.h>
#include <ctkged.h>
#define  WDW_ORG_T     20
#define  WDW_ORG_Y     180
#define  WDW_W         400
#define  WDW_H         140
#define  BACK_COLOR WHITE
void main(void)
{
  unsigned int status;
  unsigned char Datain[MAX_INPUT_DATA_SIZE];
  ctk_make_window(WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  status = ctk_ged_get_inputs(1, 1, Datain);
  if(!status)
    printf("The first byte is %X\n", Datain[0]);
  else
    printf("Error reading device input table\n");
  while(!kbhit())   /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch();      /* Remove character from console input buffer */
}
```

ctk_ged_put_outputs()

ctk_ged_put_outputs() writes the device output table (up to 128 bytes) for one Genius Bus device.

C Synopsis

```
#include <ctkged.h>
unsigned int ctk_ged_put_outputs(unsigned int hwy_number,
                                 unsigned int device_number,
                                 unsigned int data_length,
                                 unsigned char* output_buffer);
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

device_number — the Genius Bus device number (0 - 31) to which you wish to write the device output table.

data_length — the number of bytes (1 - 128) to be written to the Genius Bus device. The first byte in output_buffer will be written to the first byte of the device output table for the selected device.

output_buffer — a pointer to a buffer containing the device output data.

Output Details

No output parameters are returned.

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER
GED_FORMAT_ERROR	GED_NO_RESPONSE

Notes

To use this feature, your PCIM must be configured (with DIP switches) as device 31 on the Genius Bus, as other Genius Bus devices (such as Genius blocks) will only recognize control outputs from device 31. Also, device outputs must be enabled, either through setting the daughterboard DIP switches or by using he ctk_ged_enable_output() function.

Example: ctk_ged_put_outputs()

```
/*
** Writes 10 bytes to device output table for device 1
*/
#include <ctk.h>
#include <ctkged.h>
#define WDW_ORG_T 20
#define WDW_ORG_Y 180
#define WDW_W     400
#define WDW_H     140
#define BACK_COLOR  WHITE
void main (void)
{
  unsigned int status;
  int i:
  unsigned char DataOut[2];
  ctk_make_window(WDW_ORG_T, WDW_ORG_Y, WDW_ORG_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  /*
  ** Set up the output data:
  */
  for(i=0; i<10; i++)
  {
    Dataout[i] = 10*(i+1);
  }
  /*
  ** Send the data:
  */
  status = ctk_ged_put_outputs(1,1,10, DataOut);
  if(!status)
    printf("Outputs successfully written\n");
  else
    printf("Error writing device output table\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch(); /* Remove character from console input buffer */
}
```

ctk_ged_get_word()

ctk_ged_get_word() reads one 16-bit word from the device input table for one Genius Bus device.

C Synopsis

```
#include <ctkged.h>

unsigned int ctk_ged_get_word (unsigned int hwy_number,
                               unsigned int device_number,
                               unsigned int word_number,
                               unsigned char* input_buffer);
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

dev_number — the Genius Bus device number (0 - 31) from which you wish to read an input word.

word_number — the word number (1-64) within the device input table for the selected device that you wish to read.

Output Details

input_buffer — a pointer to a buffer of 2 bytes, where the input data will be written.

Return Codes

GED_SUCCESS
GED_INVALID_HWY
GED_FORMAT_ERROR
GED_SYSTEM_ERROR
GED_NO_DRIVER
GED_NO_RESPONSE

Example: ctk_ged_get_word()

```
/*
** Reads input word 15 from Device 1
*/
#include <ctk.h>
#include <ctkged.h>
#define WDW_ORG_T 20
#define WDW_ORG_Y 180
#define WDW_W     400
#define WDW_H     140
#define BACK_COLOR  WHITE
void main (void)
{
  unsigned int status;
  unsigned char Datain[2];
  ctk_make_window(WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H,  BACK_COLOR);
  ctk_cursor_off();
  status = ctk_ged_get_word (1, 1, 15, Datain);
  if(!status)
    printf("Word 15 = %X\n", 256*Datain[1]+Datain[0]);
  else
    printf("Error reading input word\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch(); /* Remove character from console input buffer */
}
```

ctk_ged_put_word()

ctk_ged_put_word() writes one 16-bit output word to the device output table for one Genius Bus device.

C Synopsis

```
#include <ctkged.h>

unsigned int ctk_ged_put_word (unsigned int hwy_number,
                               unsigned int device_number,
                               unsigned int word_number,
                               unsigned int data);
```

Input Details

hwy_number	the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.
device_number	the Genius Bus device number (0-31) to which you wish to write the output word.
word_number	the word number (1-64) within the device output table for the selected device that you wish to write to.
data	the data to be written to the device output word.

Output Details

No output parameters are returned.

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER
GED_FORMAT_ERROR	GED_NO_RESPONSE

Notes

To use this feature, your PCIM must be configured (with DIP switches) as device 31 on the Genius Bus, as other Genius Bus devices (such as Genius blocks) will only recognize control outputs from device 31. Also, device outputs must be enabled, either through setting the daughterboard DIP switches or by using the ctk_ged_enable_outputs() function.

Example: ctk_ged_put_word()

```
/*
** Set output word 9 for Device 1 to 0xFFFF
*/
#include <ctk.h>
#include <ctkged.h>
#define WDW_ORG_T 20
#define WDW_ORG_Y 180
#define WDW_ORG_W 400
#define WDW_H     140
#define BACK_COLOR  WHITE
void main(void)
{
  unsigned int status;
  ctk_make_window(WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  status = ctk_ged_put_word(1, 1, 9, 0xFFFF);
  if(!status)
    printf("Output successfully writen\n");
  else
    printf("Error writing output word\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch(); /* Remove character from console input buffer */
}
```

ctk_ged_get_bit()

ctk_ged_get_bit() reads one bit from the device input table for one Genius Bus device.

C Synopsis

```
#include <ctkged.h>
unsigned int ctk_ged_get_bit  (unsigned int hwy_number,
                               unsigned int device_number,
                               unsigned int bit_number,
                               unsigned char* input_buffer);
```

Input Details

hwy_number — the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.

device_number — the Genius Bus device number (0 - 31) from which you wish to read an input bit.

bit_number — the bit number (1 - 1024) within the device input table for the selected device that you wish to read.

Output Details

input_buffer — a pointer to a buffer of 1 byte, which will contain the following data upon successful function completion:
0 = the specified bit is reset
1 = the specified bit is set

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER
GED_FORMAT_ERROR	GED_NO_RESPONSE

Example: ctk_ged_get_bit()

```
/*
** Reads input bit 159 from device 1
*/
#include <ctk.h>
#include <ctkged.h>
#define WDW_ORG_T 20
#define WDW_ORG_Y 180
#define WDW_W     400
#define WDW_H     140
#define BACK_COLOR  WHITE
void main(void)
{
  unsigned int status;
  unsigned char Datain[1];
  ctk_make_window(WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  status = ctk_ged_get_bit(1, 1, 159, Datain);
  if(!status)
  {
    if(Datain[0])
      printf("Bit 159 is set.");
    else
      printf("Bit 159 is reset.");
  }
  else
    printf("Error reading input\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch(); /* Remove character from console input buffer */
}
```

ctk_ged_put_bit()

ctk_ged_put_bit() writes one output bit to the device output table for one Genius Bus device.

C Synopsis

```
#include <ctkged.h>

unsigned int ctk_ged_put_bit(unsigned int hwy_number,
                             unsigned int device_number,
                             unsigned int bit_number,
                             unsigned char data);
```

Input Details

hwy_number	the ControlView logical highway number (1 or 2) that has been assigned to the PCIM adapter using the *Configure GE PCIM* selection of the Setup/Configure menu.
device_number	the Genius Bus device number (0 - 31) to which you wish to write the output bit.
bit_number	the bit number 91 - 1024) within the device output table for the selected device that you wish to write to.
data	set to 0 to reset the specified output bit, or to any non-zero value to set the output bit.

Output Details

No output parameters are returned.

Return Codes

GED_SUCCESS	GED_SYSTEM_ERROR
GED_INVALID_HWY	GED_NO_DRIVER
GED_FORMAT_ERROR	GED_NO_RESPONSE

Notes

To use this feature, your PCIM must be configured (with DIP switches) as device 31 on the Genius Bus, as other Genius Bus devices (such as Genius blocks) will only recognize control outputs from device 31. Also, device outputs must be enabled, either through setting the daughterboard DIP switches or by using the ctk_ged_enable_outputs() function.

Example: ctk_ged_put_bit()

```
/*
** Sets output bit 15 for device 1
*/
#include <ctk.h>
#include <ctkged.h>
#define WDW_ORG_T 20
#define WDW_ORG_Y 180
#define WDW_W     400
#define WDW_H     140
#define BACK_COLOR  WHITE
void main(void)
{
  unsigned int status;
  ctk_make_window(WDW_ORG_T, WDW_ORG_Y, WDW_W, WDW_H, BACK_COLOR);
  ctk_cursor_off();
  status = ctk_ged_put_bit(1, 1, 15, 1);
  if(!status)
    printf("Output bit successfully written\n");
  else
    printf("Error writing output\n");
  while(!kbhit()) /* Pause for display until key is pressed */
        ctk_sleep(1);
  getch(); /* Remove character from console input buffer */
}
```

T

V

W

ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

As a subsidiary of Rockwell International, one of the world's largest technology companies — Allen-Bradley meets today's challenges of industrial automation with over 85 years of practical plant-floor experience. More than 11,000 employees throughout the world design, manufacture and apply a wide range of control and automation products and supporting services to help our customers continuously improve quality, productivity and time to market. These products and services not only control individual machines but integrate the manufacturing process, while providing access to vital plant floor data that can be used to support decision-making throughout the enterprise.

With offices in major cities worldwide

WORLD HEADQUARTERS
Allen-Bradley
1201 South Second Street
Milwaukee, WI 53204 USA
Tel: (1) 414 382-2000
Telex: 43 11 016
FAX: (1) 414 382-4444

EUROPE/MIDDLE EAST/AFRICA HEADQUARTERS
Allen-Bradley Europe B.V.
Amsterdamseweg 15
1422 AC Uithoorn
The Netherlands
Tel: (31) 2975/43500
Telex: (844) 18042
FAX: (31) 2975/60222

ASIA/PACIFIC HEADQUARTERS
Allen-Bradley (Hong Kong) Limited
Room 1006, Block B, Sea View Estate
28 Watson Road
Hong Kong
Tel: (852) 887-4788
Telex: (780) 64347
FAX: (852) 510-9436

CANADA HEADQUARTERS
Allen-Bradley Canada Limited
135 Dundas Street
Cambridge, Ontario N1R 5X1
Canada
Tel: (1) 519 623-1810
FAX: (1) 519 623-8930

LATIN AMERICA HEADQUARTERS
Allen-Bradley
1201 South Second Street
Milwaukee, WI 53204 USA
Tel: (1) 414 382-2000
Telex: 43 11 016
FAX: (1) 414 382-2400

Publication 6190-6.5.26 November 1992

PN 504669101